LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Rajapinnan regressiotestauksen automaatiojärjestelmän valinta

Jukka-Pekka Manninen

2017 Laurea

Laurea-ammattikorkeakoulu

Rajapinnan regressiotestauksen automaatiojärjestelmän valinta

Jukka-Pekka Manninen
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Marraskuu, 2017

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Jukka-Pekka Manninen

Rajapinnan regressiotestauksen automaatiojärjestelmän valinta

Vuosi 2017 Sivumäärä 35

Tutkimuksen tarkoitus oli löytää kohdeyritykselle sopiva avoimen lähdekoodin järjestelmä kohdeyrityksen tuotteiden tarjoamien REST-rajapintojen regressiotestaamisen automatisointiin. Testiautomaation käyttöönoton tavoitteena oli parantaa kohdeyrityksen tuotteiden laatua ja toiminnan tehokkuutta lisäämällä testien käyttöä. Tämän oletettiin vähentävän kehitystyössä tapahtuvia tahattomia muutoksia rajapintojen toiminnassa ja näiden korjaamiseen tarvittavaa tuottamatonta takuutyötä. Tutkimusongelma oli selvittää, onko kohdeyritykselle sopivia testiautomaatiojärjestelmiä ylipäänsä olemassa ja mikä niistä on sekä teknisesti että muilta ominaisuuksiltaan sopivin kohdeyrityksen käyttöön.

Järjestelmien sopivuuden vertailemiseksi muutamaa sopivimmalta vaikuttanutta vaihtoehtoa kokeiltiin käytännössä. Vertailun tasapuolisuuden vuoksi ennen järjestelmien valintaa asetettiin joukko kriteereitä, jotka kohdeyritykselle sopivan järjestelmän tulisi täyttää. Kokeista laadittiin muistiinpanoja ja havaintoja analysoitiin laadullisen luokittelun menetelmällä. Kolmesta sopivimmalta vaikuttavasta järjestelmästä kahta pystyttiin kokeilemaan käytännössä. Kolmannen järjestelmän kohdalla testaaminen epäonnistui järjestelmän asennuspaketin virheen vuoksi. Molemmat testatut järjestelmät olisivat sopineet kohdeyrityksen käyttöön, mutta toinen näistä täytti kriteerit paremmin. Lopputuloksena saatiin suositus kohdeyritykselle käyttöönotettavasta järjestelmästä. Myös järjestelmän käyttöönoton vaikuttavuuden seuraamista suositeltiin.

Sopivimman testiautomaatiojärjestelmän löytäminen ja käyttöön ottaminen on kohdeyritykselle merkittävä tehokkuustekijä. Tutkimuksessa käytettyä menettelytapaa voidaan myös soveltaa yleisemminkin yrityksen liiketoimintaa tukevien tietojärjestelmien valintaan.

Asiasanat: Rajapinta, ohjelmistotestaus, testiautomaatio, regressiotestaus, REST

Laurea University of Applied Sciences

Degree Programme in Business Information Technology
Bachelor of Business Administration

Jukka-Pekka Manninen

Abstract

Selecting Test Automation Framework for Regression Testing of an Interface

Year 2017 Pages 35

The objective of this Bachelor's thesis was to discover an open source test automation framework suitable to automate regression testing RESTful interfaces at the commissioning company. Test automation was intended to improve product quality and efficiency of operations at the commissioning company by increasing the usage of regression testing. This was expected to reduce the frequency of unintended changes in the behaviour of interfaces and unbillable work needed to revert these changes. The research problem was to investigate whether suitable test automation frameworks exist at all and if so, which one is best suited for the commissioning company.

To compare the suitability a few apparently most fitting frameworks were put to test in practise. For unbiased comparison a set of criteria to define a suitable framework for the commissioning company were placed. Notes were taken during these tests and their observations were analysed through qualitative categorization. Two out of the three apparently most suitable frameworks were successfully tested. Testing of the third system failed because the published installation package was defective. Although both successfully tested frameworks were found suitable for the commissioning company one of them fulfilled the placed criteria better. The result was a recommendation for the commissioning company to deploy specified test automation framework. Monitoring the effectiveness of the deployment was also recommended.

Discovery and deployment of the best suited test automation framework will be a major influence on efficiency at the commissioning company. The methodology used in this study can be adapted to more generally examine the suitability of information systems to support production work.

Keywords: Interface, software testing, test automation, regression testing, REST

Sisällys

1 Johdanto

Kohdeyritys on merkittävä kotimainen sähköisen kaupankäynnin palvelujen tuottaja. Sen merkittävimpiin asiakkaisiin sisältyy useita suuria kotimaisia yrityksiä monilta eri liiketoiminta-aloilta. Siksi kohdeyritykselle on erityisen tärkeää tarjota luotettavaa ja laadukasta palvelua. Yrityksen toimittamien tuotteiden hyvä laatu ja mahdollisimman pieni virheiden määrä on yksi tärkeä osa tätä laatua.

Kohdeyrityksen pääasiallisena ohjelmistotuotannollisena tuotteena ovat verkkokauppajärjestelmät. Niiden päälle rakennetuista verkkokaupoista löytyy niin kuluttajakauppaan, yritysten väliseen kaupankäyntiin kuin julkishallinnon sopimushankintojen tekemiseen tarkoitettuja kauppoja. Muutama verkkokauppa on jopa suunnattu useammalle näistä kohderyhmistä samaan aikaan.

Suurilla yrityksillä verkkokauppa harvoin on ainoa kaupankäynnin muoto, vaan liiketoimintaan kuuluu myös esimerkiksi kivijalkakauppoja tai suurten erien tilausmyyntiä yritysasiakkaille. Tyypilliset verkkokauppa-alustat eivät sovellu tällaisen logistiikkaketjun hallintaan, ja toisaalta siihen tehtävään kehitetyt järjestelmät taipuvat huonosti verkkokaupan pyörittämiseen. Siksi tiedonvaihto näiden eri tarkoituksia palvelevien järjestelmien välillä on saatava asiakkaan tarpeet täyttävän joustavaksi, luotettavaksi ja nopeaksi.

Tiedonsiirron voisi hoitaa työntekijä, joka lukee tiedot yhdestä järjestelmästä ja kirjaa ne toiseen. Itse asiassa liiketoiminnan tietokoneistumisen alkuaikoina näin usein toimittiinkin. Tietokoneiden suorituskyky on vuosikymmenien aikana monisatakertaistunut ja hinta romahtanut samalla kun käsiteltävän tiedon määrä on kasvanut voimakkaasti, joten nykyään on tietojen siirto järjestelmien välillä automatisoitava jo pienissäkin yrityksissä kilpailukyvyn säilyttämisen vuoksi.

Kohdeyritys onkin profiloitunut tällaisten tiukasti asiakkaan liiketoimintajärjestelmiin integroituvien verkkokauppojen toimittajana. Niitä kohtia, joissa järjestelmät vaihtavat tietoa keskenään, kutsutaan rajapinnoiksi. Koska niissä kohtaa kahden erillisen kehitysryhmän työ, eivätkä nämä ryhmät välttämättä koskaan kohtaa toisiaan, vaan tietoa saatetaan vaihtaa vain sähköpostilla, on näissä kohdissa kohonnut riski saada tapahtumaan virheitä.

Virheitä ei toisaalta voida inhimillisessä työssä koskaan kokonaan välttää. Siksi järjestelmiä ja rajapintoja testataan, jotta väistämättömät virheet voidaan havaita ja korjata mahdollisimman aikaisessa vaiheessa. Tällöin niiden kustannusvaikutus jää pienemmäksi kuin jos virheet huomattaisiin vasta todellisessa käytössä. Kun rajapinta on tarkoitettu kahden koneellisen tietojärjestelmän väliseen tiedonvaihtoon, niin myös sitä testaamaan voidaan asettaa kone.

Tutkimuksen tavoitteena onkin löytää kohdeyrityksen tarpeeseen sopiva testiautomaatiojärjestelmä. Kohdeyrityksessä on tunnistettu tarve helpottaa ja lisätä rajapintojen regressiotestausta, koska nykyisellään sitä tehdään riittämättömästi ja virheitä joudutaan korjaamaan jälkikäteen. Parhaaksi keinoksi tähän on kohdeyrityksen sisällä todettu automaattisen testaamisen lisääminen. (N.N. 2015.)

2 Työn lähtökohdat

Kohdeyrityksellä on tarve automatisoida REST-rajapinnan regressiotestausta. Tällä hetkellä käytössä oleva työkalu vaatii paljon manuaalista työtä. Lisäksi se koetaan hyvin vaikeaselkoiseksi. Näistä syistä regressiotestausta tehdään huomattavasti vähemmän kuin tarve olisi. (N.N. 2015.)

Regressiotestauksen vähäisyyden vuoksi rajapinnoissa voi päästä tapahtumaan tahattomia muutoksia, jotka vaarantavat järjestelmän oikean toiminnan. Tuotantoon asti päästessään nämä virheet aiheuttavat paljon kallista työtä, kun korjaus on saatava valmiiksi nopeassa aikataulussa. Siksi regressiotestauksen helpottamiseen on kohdeyrityksellä selkeä intressi. (N.N. 2015.)

Mikäli tarkoitukseen sopivaa työkalua ei tutkimuksessa löydy, on kohdeyrityksellä tahtoa kehittää sellainen olemassa olevaan avoimen lähdekoodin työkaluun perustuen. Näin ollen myös kyseinen työkalu tulisi todennäköisesti julkaistuksi avoimen lähdekoodin lisenssin alla. Kohdeyritys on toiminut näin myös aikaisemmin, kun tarpeeseen soveltuvaa sovellusta ei löytynyt. Silloin myös koko maailmanlaajuinen ohjelmistokehitysyhteisö hyötyisi uuden avoimen testaustyökalun muodossa. (N.N. 2015.)

Opinnäytetyön tutkimusongelma voidaan kiteyttää kahteen kysymykseen, joista jälkimmäinen on merkityksellinen vain, jos ensimmäisen vastaus on myöntävä.

- Onko olemassa avoimen lähdekoodin lisenssillä julkaistuja testiautomaatiojärjestelmiä, jotka voisivat soveltua kohdeyritykselle REST-rajapinnan automaattiseen regressiotestaukseen?
- Mikäli on, mikä näistä järjestelmistä soveltuu parhaiten kohdeyrityksen käyttöön?

Opinnäytetyön aihetta rajaavat voimakkaasti toimeksiantajan tarpeet. Tutkimuksen kohteena tulee olemaan testauksen automatisointijärjestelmien soveltuvuus toimeksiantajan organisaation käyttöön. Toimeksiantaja haluaa lisäksi järjestelmiä arvioitavan nimenomaan REST-rajapinnan regressiotestauksen kannalta. Järjestelmän käyttöönotto tai sellaisen kehittäminen, mikäli sopivaa valmista vaihtoehtoa ei löydy, eivät sisälly tähän tutkimukseen. (N.N. 2015.)

Kohdeyrityksellä ei kuitenkaan ole resursseja kokeilla jokaista mahdollista järjestelmä todellisessa tuotantokäytössä. Niinpä niiden soveltuvuutta on arvioitava muilla keinoin, vaikka näin

ei olekaan mahdollista huomioida jokaista yksityiskohtaa joka vaikuttaa järjestelmän soveltuvuuteen. Järjestelmistä täytyy siitä huolimatta selvittää useampia soveltuvuuden osa-alueita. Näitä osa-alueita käsitellään tarkemmin tutkimuksen suorittamisvaiheessa, mutta alustavien keskustelujen perusteella niitä voisivat olla esimerkiksi:

- Testitapausten laatiminen.
- Järjestelmän integroituvuus muihin toimeksiantajan järjestelmiin.
- Järjestelmän tuki REST-rajapintojen testaamiseen.

3 Keskeiset käsitteet

Avoimen lähdekoodin lisenssillä tai vapaalla lisenssillä tarkoitetaan lisenssiä, joka ei aseta rajoituksia lisensoidun kohteen käytölle, tutkimiselle, levittämiselle tai kehittämiselle. Erityisesti se ei tarkoita, että ohjelmasta, sen käytöstä tai käytön tukipalveluista ei voisi pyytää korvausta. Esimerkkejä laajalti käytetyistä vapaista lisensseistä ovat GNU General Public Licence ja Creative Commonsin lisenssiperhe. (Mitä vapaat ohjelmistot ovat 2017.)

Software as a Service (SaaS), suomeksi ohjelmisto palveluna, tarkoittaa mallia jossa hankitaan käyttöoikeus palveluntarjoajan omilla palvelimillaan ylläpitämään sovellukseen. Tällaista sovellusta käytetään yleisimmin verkkoselaimella ja niistä maksetaan kuukausittain mahdollisesti käyttäjien määrästä riippuva maksu. Jotkut yritykset tarjoavat myös avoimen lähdekoodin sovelluksia käyttöön SaaS-palveluna, jolloin palvelun ostajan ei tarvitse itse huolehtia sovelluksen asentamisesta, konfiguroimisesta tai päivittämisestä. (Mikä on SaaS 2010.)

Paketinhallintajärjestelmä on käyttöjärjestelmän osa, jonka tehtävä on huolehtia sovellusten asentamisesta tietokoneeseen. Se pitää huolen, että kaikki sovelluksen tiedostot päätyvät oikeille paikoilleen, sovellus saa tarvitsemansa käyttöoikeudet tietokoneen resursseihin ja sovelluksen käynnistämiseen tarvittavat kuvakkeet tai vastaavat ovat käyttäjien löydettävissä. Paketinhallintajärjestelmä voi myös poistaa sovelluksen, ja monissa käyttöjärjestelmissä se myös hakee ja asentaa käyttöjärjestelmän ja sovellusten päivitykset.
(Paketinhallintajärjestelmä 2009.)

3.1 Ohjelmistokehitys

Ohjelmistokehitys tarkoittaa prosessia tai työtä, jonka kautta luodaan kokonaan uusi sovellus tai lisätään vanhaan sovellukseen uusia ominaisuuksia. Opinnäytetyössä sivutaan monia ohjelmistokehityksen aihealueeseen kuuluvia käsitteitä, jotka eivät suoranaisesti kuulu tutkimuksen varsinaiseen kohdealueeseen. Niiden ymmärtäminen on kuitenkin tarpeen itse tutkimuksen tavoitteiden ja suorittamisen ymmärtämiseksi.

JIRA on Atlassian Inc.:n valmistama tehtävienhallintaohjelmisto. Sillä voidaan hallinnoida ohjelmistokehityksen kaikkia tehtäviä projektin ensimmäisten määrittelyjen laatimisesta sovelluksen elinkaaren viimeisiin virheenkorjauksiin asti. JIRAssa halutuille tehtäville tehdään

tiketit, joihin kirjataan tehtävän suorittamisen kannalta keskeiset tiedot. Tiketti voidaan sitten antaa vuorollaan eri käyttäjien vastuulle kutakin tehtävän suorittamisen vaihetta varten. (Eduix 2009.)

Tiedonsiirtokäytäntö tai -protokolla tarkoittaa niitä sääntöjä, joiden avulla kaksi tietoverkon kautta yhteydessä olevaa tietokonetta voivat vaihtaa tietoja keskenään. Protokolla voi olla hyvin erikoistunut kuten tiedostojen kopioimiseen tietokoneelta toiselle tarkoitettu FTP (File Transfer Protocol) tai yleisluontoinen kuten alun perin verkkosivujen lataamiseen suunniteltu HTTP (HyperText Transfer Protocol). Jälkimmäinen on joustavuutensa vuoksi muodostunut de facto standardiksi kaikenlaisessa tietoliikenteessä tietokoneiden välillä.

Continuous Integration eli jatkuva integraatio tarkoittaa prosessia, jossa koko kehitettävää järjestelmää koostetaan yhteen jatkuvasti kehitystyön edetessä. Koostaminen voi tapahtua jokaisen muutostehtävän valmistumisen jälkeen tai ajastetusti esimerkiksi joka yö. Tämän prosessin toteuttamiseen on kehitetty useita järjestelmiä, kuten Atlassian Inc.:n Bamboo ja avoimen lähdekoodin lisenssillä julkaistu Jenkins. (Poimala & Tolvanen 2013.)

Virtualisointijärjestelmä on sovellus, jonka avulla yhden fyysisen tietokoneen voi jakaa moneksi virtuaaliseksi tietokoneeksi. Näin yhdellä tietokoneella voidaan ajaa useampia sovelluksia niiden häiritsemättä toinen toistaan. Virtualisointijärjestelmän avulla kerran asennetusta järjestelmästä voidaan myös käynnistää useita identtisiä kopioita, mikä helpottaa järjestelmän monistamista suorituskyvyn parantamiseksi tai saman järjestelmän asentamista useiden eri asiakkaiden ympäristöihin. Tässä tutkimuksessa on käytetty uuden polven kevyempää virtualisointisovellusta nimeltä Docker. (What is Docker? 2014.)

Apache Maven on Java-ohjelmistoprojektin käännösprosessin hallintaan kehitetty työkalu. Se perustuu keskitettyyn konfiguraatiotiedostoon, jonka perusteella se osaa suorittaa sovelluksen kääntämiseksi tarpeelliset askelet. Tyypillisiä askeleita ovat tarvittujen valmiskirjastojen lataaminen näiden jakeluun tarkoitetusta verkkopalvelusta, ihmisen luettavan lähdekoodin kääntäminen koneen suoritettavaksi tavukoodiksi ja tuotosten pakkaaminen jakelukelpoiseen tiedostoon. (Apache Maven Project 2017.)

3.2 Rajapinnat

Rajapinta tarkoittaa ohjelmistokehityksessä ohjelman tai järjestelmän osan toiminnallista lupausta tai sopimusta. Se määrittelee mitä tietoja kyseinen osa käsittelee ja mitä se kyseisillä tiedoilla tekee, mutta ei käytännön toteutusta millä tavalla se tämän käsittelyn suorittaa. Tämä mahdollistaa ohjelman osan toteutuksen muuttamisen ilman vaikutuksia osaa käyttäviin muihin osiin, kunhan sen rajapinta ei muutu. Rajapinta voi olla tarkoitettu myös kahden tai useamman erillisen, jopa eri organisaation kuten yrityksen, järjestelmien väliseen tiedonvaihtoon. Silloin sitä kutsutaan integraatiorajapinnaksi. (Jia 2000, 12.)

Rajapinnaksi voidaan käsittää myös esimerkiksi kaupan kassalla oleva korttimaksupääte. Asiakkaat osaavat laittaa maksukortin lukijaan ja syöttää PIN-koodinsa ilman erityistä keskittymistä suoritukseen. Mutta jos tässä rajapinnassa tapahtuisikin odottamaton muutos, esimerkiksi maksupääte haluaisikin PIN-koodin ennen kuin kortin laittaa lukijaan, monet käyttäjät menisivät hämilleen eivätkä heti osaisi reagoida tilanteeseen.

Integraatiorajapinnat toteutetaan käytännössä tietoverkon ylitse. Nykyään kommunikaatioprotokollana on jokseenkin aina HTTP-protokollan mukaisesti lähetetyt sanomat, vaikka menneisyydessä myös esimerkiksi tiedostojen siirtoa FTP-protokollalla on käytetty. Sanomien määrittelyyn ovat vakiintuneet XML-kieleen nojautuva SOAP- ja vapaamuotoisempi REST-metodologia.

REST (tai ReST) tulee englanninkielisistä sanoista Representational State Transfer. Sillä tarkoitetaan verkkorajapintaa, joka on jäsennelty palvelun sisältämien resurssien mukaan sen tarjoamien toimenpiteiden sijaan. Tällainen palvelu myös palauttaa resurssien lisäksi kuvauksen resurssille mahdollisista toimenpiteistä. RESTful on termin adjektiivimuoto. SOAP (Simple Object Access Protocol) on RESTiä aikaisempi ja tiukemmin määritelty standardi (XML Soap). (Richardson & Ruby 2007, 13.)

4 Ohjelmistotestaus ja testiautomaatio

Ihmiset tekevät jatkuvasti virheitä ajatuksissaan, puheessaan, kuunnellessaan tai mitä tahansa ovatkin milloin tekemässä. Suurin osa näistä virheistä on mitättömiä pieniä lipsahduksia, jotka korkeintaan herättävät pienen naurunhörähdyksen, jos joku ylipäänsä huomaa niitä laisinkaan. Mutta joskus virheillä voi olla vakavia seurauksia. (Mathur 2008, 4.)

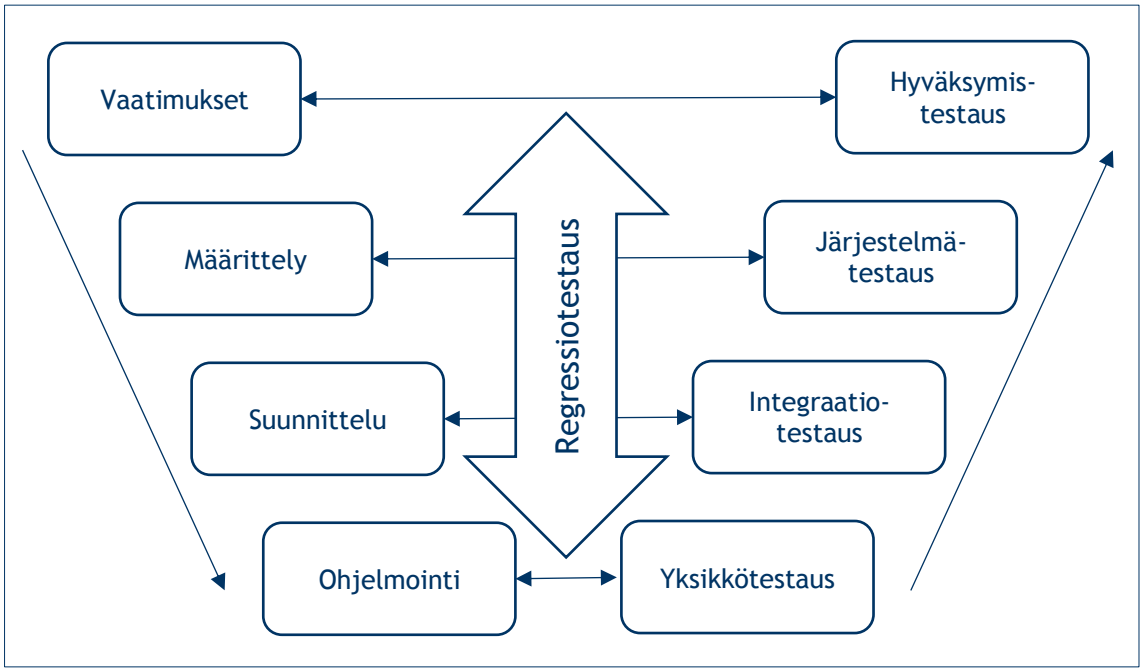
Siksi ihmisten työn tuloksia, kuten tietokoneita ja ohjelmistoja, joita tietokoneilla suoritetaan, testataan. Testaamisen tehtävä on varmistaa, että työn lopputulos on se kokonaisuus mikä alun perin haluttiin luoda ja se toimii niin kuin sen haluttiin toimivan. (Kasurinen 2013, 10.)

Testaamista voidaan jaotella eri tavoin. Kehitystyön vaiheen mukaan testaaminen voidaan jakaa pienten osien yksikkötestaukseen, näiden yhteistoiminnan integraatiotestaukseen, kokonaisuuden järjestelmätestaukseen ja asiakkaan vaatimusten toteutumisen tarkistavaan hyväksyntätestaukseen. Näiden tasojen suhteet on kuvattu kuviossa 1. Testausmenetelmän mukaan voidaan puhua muun muassa käyttöliittymää tulevilla käyttäjillä testauttavista käytettävyyskokeista, järjestelmän tietyllä syötteellä antamia tuloksia tarkastelevista musta laatikko-testeistä tai järjestelmän selviytymistä suurista tieto- tai käyttäjämääristä tarkastelevasta kuormitustestauksesta. (Kasurinen 2013.)

Regressiotestauksen tavoite on estää testattavan järjestelmän toiminnan tahaton muuttuminen kehitystyön aikana. Se ei siis ole varsinaisesti tiettyyn projektin vaiheeseen nivoutuvaa

testaamista eikä tarkoita mitään tiettyä testaamisen menetelmää. Koska järjestelmän toiminta testataan regressiotestausmielessä monia kertoja kehitystyön aikana, se on hyvä kohde testitoiminnan automatisoinnille. (Kasurinen 2013, 54-55.)

Kohdeyrityksessä yksittäisen kehitystehtävän toteuttaminen seuraa melko hyvin projektin V-mallia, joka on esitelty kuviossa 1 (Kasurinen 2013, 51). Yksikkötestaus on automatisoitu ja järjestelmätestausta käyttöliittymän osalta hoitavat palkatut testaajat. Hyväksymistestaus on asiakkaan oikeus ja velvollisuus, sillä heidän vastuullaan on hyväksyä valmis työ toimitetuksi. Erilliselle integraatiotestaukselle ei ole kohdeyrityksessä toistaiseksi koettu tarvetta. Regressiotestausta tehdään suorittamalla testit uudestaan läpi kaikkien tasojen aina, kun tehty muutos on läpäissyt alempien tasojen testit.

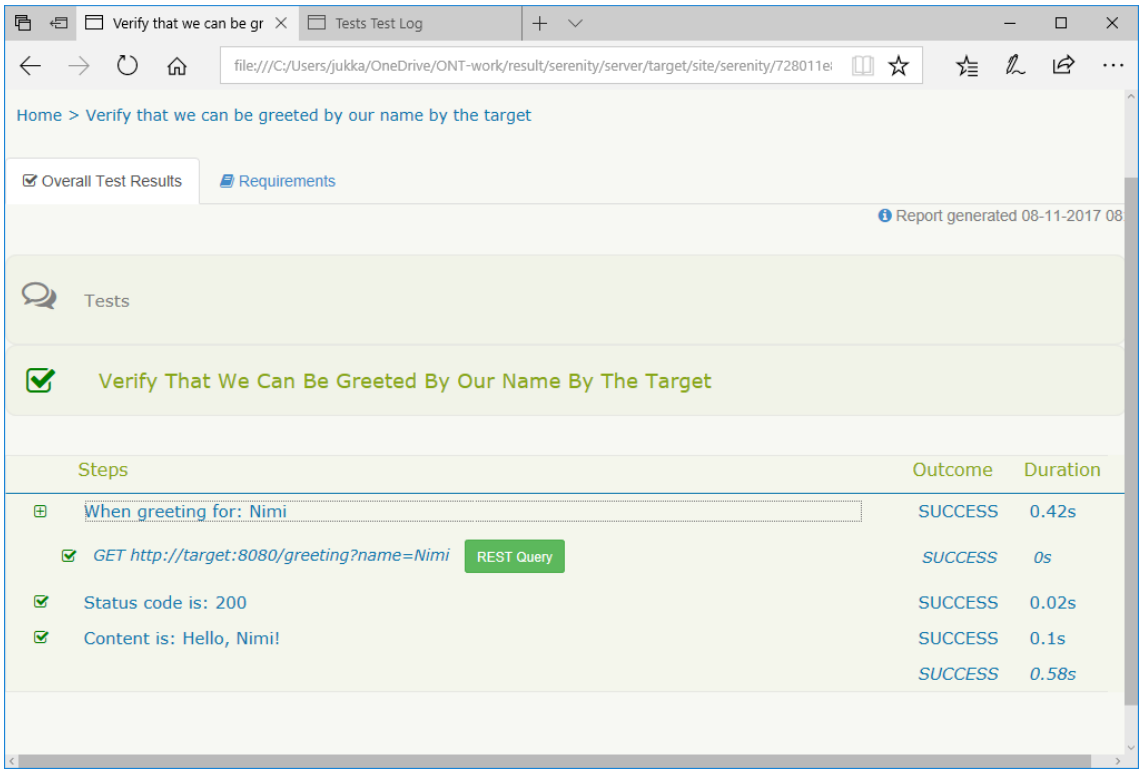


Kuvio 1: Projektin V-malli.

Testausautomaatio tai testiautomaatio tarkoittaa testauksen suorittamisen siirtämistä ihmiseltä tähän tarkoitukseen valmistetun laitteen tai ohjelmiston tehtäväksi. Sen tavoitteena voi olla vähentää testaukseen tarvittavan työvoiman määrää tai tehostaa ajankäyttöä kehityksen edistämiseen, kun kone voi suorittaa testejä öisin kehittäjien ollessa kotona nukkumassa. (Kasurinen 2013, 60.)

Tähän tehtävään kehitettyä laitetta tai ohjelmistoa kutsutaan testiautomaatiojärjestelmäksi. Monilla organisaatioilla on sisäisesti kehitettyjä testiautomaatiojärjestelmiä, mutta niitä on saatavana myös kaupallisesti ja avoimen lähdekoodin lisensseillä. Testiautomaatiota käytetään usein yhdessä jatkuvan integraation kanssa, jolloin järjestelmän uuden version koostamisen jälkeen sille suoritetaan automaattitestit. Järjestelmä luo testien tuloksista kehittäjien

käyttöön raportin, joka voi näyttää esimerkiksi sellaiselta kuin kuviossa 2. (Kasurinen 2013, 68.)



Kuvio 2: Serenityn lokitiedot ovat selkeät ja yksinkertaiset.

Integraatiorajapinnan tapauksessa regressiotestauksen automatisointi järjestelmätestaustasolle asti voi tuoda erityistä etua manuaaliseen testaukseen verrattuna. Koska integraatiorajapinta on tarkoitettu järjestelmien väliseen viestintään, niiden manuaalinen testaaminen vaatii samaa hyvää teknistä asiantuntemusta kuin automaattisten testien määritteleminen. Siihen tarvitaan siis käytännössä ohjelmoijan taidot omaava tekijä. Lisäksi teknisten rajapintojen testaaminen manuaalisesti vaatii runsaasti työtä siihen nähden, paljonko merkityksellistä tietoa rajapinnan kautta kulkee.

5 Tutkimusmenetelmät

Opinnäytetyöhön voidaan soveltaa kolmea lähestymistapaa. Yleensä yksi näistä on selkeästi pääasiallinen lähestymistapa, mutta tutkimuksen kohteesta ja laajuudesta riippuen myös muita menetelmiä voidaan soveltaa niiltä osin kuin soveltuvia käyttökohteita löytyy.

5.1 Määrällinen tutkimus

Määrällinen tutkimus keskittyy mitattavissa oleviin ilmiöihin. Tällaisessa tutkimuksessa usein lähestytään suurta ihmisjoukkoa standardoiduin kyselyin, jolloin vastauksia ja niiden jakautumista voidaan käsitellä tilastollisin menetelmin. (Saunders, Lewis & Thornhill 2009, 25.)

Tähän tutkimuksen määrällinen tutkimusmenetelmä sopii huonosti kahdesta syystä. Ensinnäkin ihmisjoukko, jota tutkimus koskee, on erittäin pieni. Siksi tuloksia ei voida käsitellä tilastollisesti eikä ainakaan saavuttaa minkäänlaista luotettavuutta. Toiseksi tietyn sovelluksen soveltuvuus juuri tietyn organisaation käyttöön on hankala esittää puhtaasti numeraalisessa muodossa, joten tutkimuksen tätäkään osaa ei voida käsitellä ainoastaan määrällisin menetelmin. Määrälliset menetelmät otetaan käyttöön vain, jos tutkimuksessa tulee vastaan tietoa, jonka käsittelyyn ne ovat paras väline.

5.2 Laadullinen tutkimus

Laadullinen tutkimus keskittyy lisäämään ymmärrystä aihealueestaan sen sijaan että yrittäisi selittää miksi asiat tapahtuvat niin kuin tapahtuvat. Laadulliset menetelmät sopivat erityisesti tutkimuksiin, joissa tutkittavat ilmiöt eivät sovi numeroilla ilmaistavaksi tai tutkimus koskettaa pientä määrää ihmisiä. Suuria tietomääriä laadullisilla menetelmillä on vaikea käsitellä, koska tämä yksinkertaisesti vaatisi valtaisan määrän työtä. (Saunders ym. 2009, 25-26.)

Tässä opinnäytetyössä yllämainitut ehdot täyttyvät. Siksi tutkimuksessa tullaan käyttämään pääasiassa laadullisia menetelmiä. Määrälliset menetelmät otetaan kuitenkin käyttöön, mikäli jokin osa-alue osoittautuu soveltuvaksi siten käsiteltäväksi.

5.3 Projektihanke

Opinnäytetyössä on projektihankkeen piirteitä. Toimeksiantajan tavoitteena on ottaa käyttöön toimintaa merkittävästi tehostava testiautomaatiojärjestelmä. Koko projekti on kuitenkin työmäärältään liian suuri opinnäytetyöksi. Projektin ensimmäinen vaihe eli mahdollisten valmiiden järjestelmien kartoittaminen ja niiden soveltuvuuden selvittäminen on laajuudeltaan riittävä. Tähän tarkoitukseen sopii parhaiten laadullinen tutkimus.

Tähän tarkoitukseen sopii parhaiten laadullinen tutkimus. Määrälliset menetelmät saavuttavat luotettavuutta tutkittavien kohteiden lukumäärän lisääntyessä. Kohdeyrityksessä valittava järjestelmä menee noin kahdenkymmenen henkilön käyttöön ja mahdollisesti soveltuvia järjestelmiä on tutkijan arvion mukaan löydettävissä muutamia. Nämä lukumäärät eivät kunnolla riitä määrällisen käsittelyn lähtökohdaksi.

5.4 Validiteetti ja reliabiliteetti

Validiteetti ja reliabiliteetti ovat tutkimuksen oikeellisuuden ja luotettavuuden mittareita. Ollakseen pätevä tutkimuksen tulee saavuttaa korkea taso molemmissa. Validiteetti tarkastelee sitä, tuottavatko tutkimuksen valitut mittarit sitä tietoa, jota niillä on ollut tarkoitus mitata.

Validiteetti on riippuvainen reliabiliteetista. Jolleivat käytetyt menetelmät tuota luotettavaa tietoa, ei tutkimuskaan voi tuottaa luotettavaa tietoa. Toisinpäin riippuvuutta ei ole; tarkat

ja toistettavat tuloksetkin voivat mitata aivan eri asiaa kuin haluttiin. (Cohen, Morrison & Manion 2007, 133.)

5.4.1 Validiteetti

Cohen ym. toteavat, että jos tutkimus ei ole validi, se on arvoton (Research Methods in Education 2007, 133). Tutkimuksen validiteetti mittaa sitä, kuinka hyvin valitut menetelmät mittaavat niitä asioita mitä tutkimuksella halutaan mitata. Validiteetti ei siis ole ominaisuus, joka tutkimuksella joko on tai ei ole. Näin on koska tutkimustyössä niin kuin missään ihmisen toiminnassa ei koskaan voida saavuttaa täydellistä virheettömyyttä. (Cohen ym. 2007, 133.)

Määrällisessä tutkimuksessa validiteetin arviointi perustuu usein tutkimusaiheen yleisesti hyväksyttyjen menetelmien käyttöön. Laadullisessa tutkimuksessa validiteetin arviointi ei ole yhtä selkeää, vaan se perustuu esimerkiksi lähtötietojen luotettavuuden ja tutkijan objektiivisuuden arviointiin. (Cohen ym. 2007, 133-134.)

5.4.2 Reliabiliteetti

Tutkimuksen reliabiliteetti mittaa sitä, kuinka tarkasti ja toistettavasti valitut menetelmät tuottavat tietoa. Reliabiliteetti ei ota kantaa siihen, onko tämä tieto juuri sitä mitä tutkimuksella tavoiteltiin. Reliabiliteetin käytännön merkitys on erilainen kvalitatiivisessa ja kvantitatiivisessa tutkimuksessa. (Cohen ym. 2007, 146.)

Kvantitatiivisessa tutkimuksessa reliabiliteetti tarkoittaa, että tulokset ovat stabiileja ja toistettavia. Tällä tarkoitetaan, että jos tutkimus toistetaan samanlaisilla parametreilla hankituilla lähtötiedoilla, saadaan samanlaisia tuloksia. (Cohen ym. 2007, 146.)

Kvalitatiivisessa tutkimuksessa reliabiliteetin määritelmä ja jopa koko käsitteen käyttökelpoisuus on jonkin verran kiisteltyä. Vaikka kvalitatiivisen tutkimuksen lähtötiedot eivät olekaan täydellisesti toistettavia, voidaan tutkimuksessa silti toistaa menetelmiä ja tiedonhankinnan parametreja. (Cohen ym. 2007, 148.)

Kvalitatiivisen tutkimuksen reliabiliteettia voidaan arvioida siltä pohjalta, olisiko tutkija tehnyt samat havainnot ja johtopäätökset, jos olisi tarkastellut tutkimuskohteita eri aikaan tai eri paikassa tai kiinnittänyt huomiota eri ilmiöihin. Lisäksi voidaan vielä arvioida, olisiko vastaavat tiedot omaava tutkija tehnyt samoista ilmiöistä vastaavat johtopäätökset. (Cohen ym. 2007, 148.)

5.5 Tutkimusmenetelmän valinta

Opinnäytetyö voi olla muodoltaan laadullinen tutkimus, määrällinen tutkimus tai projektihanke. Näistä ehkä puhtaasti määrällinen tutkimus on harvinaisin ja sitä käytetään lähinnä luonnontieteissä ja joillakin humanistisen tutkimuksen alueilla. Määrällisiä menetelmiä

voidaan kyllä käyttää laadullisen tutkimuksen tukena tai projektihankkeessa perustelemaan tehtäviä valintoja. (Manninen 2015.)

Projektihankkeiden ongelma opinnäytetyönä on usein hankkeen laajuus. Projekti on yksinkertaisesti liian suuri tai pitkäkestoinen jotta siitä voisi järkevästi tehdä opinnäytetyön. Moneen projektiin kuitenkin kuuluu etenkin suunnitteluvaiheessa tiedonhankintatarpeita, jotka sopivat opinnäytetyön aiheeksi. Silloin lopputuloksena on tyypillisesti laadullinen tutkimus. (Manninen 2015.)

Tutkimuksen kohde on juuri tällainen projektityön suunnitteluun kuuluva selvitys. Kohdeyritys tarvitsee työkalun REST-rajapinnan regressiotestaukseen. Mikäli sopivaa valmista avoimen lähdekoodin työkalua ei löydy, on kohdeyrityksellä halukkuutta luoda sellainen itse. Kuten projektihankkeiden kohdalla yleensäkin, on koko projekti liian suuri opinnäytetyönä suoritettavaksi. Siksi tutkimus keskittyy kartoittamaan valmiit sovellukset ja tutkimaan niiden soveltuvuutta kohdeyrityksen käyttöön. (Manninen 2015.) (NN 2015.)

5.6 Tutkimuksen empiirinen osuus

Järjestelmien vertailuun käytettävät kriteerit tulee selvittää ennen itse vertailua. Alustavassa keskustelussa on tullut esille ainakin testitapausten laatimisen helppous, järjestelmän tuki REST-rajapinnan testaamiselle ja mahdollisuus integroida järjestelmä muihin kohdeyrityksen järjestelmiin (Manninen 2015). Nämä kriteerit tarkentuvat itse tutkimuksen aikana.

Ohjelmistoja on pystyttävä vertailemaan valittujen kriteerien kautta. Jotkin kriteerit voivat soveltua määrälliseen käsittelyyn. Esimerkiksi tietyn testitapauksen suoritusaika on sellainen muuttuja, josta on helppo eristää muiden tekijöiden kuin tutkittavan ohjelmiston vaikutus pois. Sen sijaan toiset tekijät voivat olla hyvinkin vaikeita kvantifioida. Kuinka määritellään testitapauksen laatimisen helppous? Tämä on vieläpä hyvin subjektiivinen aihe, yksi kehittäjä voi suosia graafista testitapauksen määrittelyä, kun toisen mielestä tekstimuotoinen määritelmä on selkeämpi.

Testeistä saatavan aineiston laadun vuoksi myös testien tulokset on parasta käsitellä laadullisen analyysin kautta. Analyysin luotettavuutta voi parantaa triangulaation avulla. Esimerkiksi aikaisemmin määritetyille kriteereille voidaan valita useampia mittareita, joiden kautta kriteerin täyttymistä arvioidaan. Näin huomataan, jos eri mittarit antavat samalle kriteerille erisuuntaisia tuloksia. (Cohen ym. 2007.)

Laadulliseen analyysiin ei ole olemassa samanlaisia selkeitä ja täsmällisesti määriteltyjä menetelmiä kuin määrälliseen analyysiin. Analyysimenetelmän valinnassa täytyy kiinnittää huomiota sen soveltuvuuteen tutkimuksen tarkoitukseen. Tämän vuoksi tutkijan täytyy aluksi selkeästi päättää, mitä haluaa analyysillään saada tutkimuskohteesta selville (Cohen ym. 2007,

461-462). Tutkimuksen kohdalla tavoitteena on löytää tutkittavien testiautomaatiojärjestelmien ominaisuuksien erot ja yhteneväisyydet kohdeyrityksen tarpeiden kanssa.

Tämän jälkeen valitaan tarkemmin menetelmä tai menetelmät, joilla tulokset analysoidaan. Sovellusten vertailuun kriteerien kautta sopiva menetelmä riippuu paitsi itse kriteereistä, myös siitä suorittaako itse testit sama henkilö joka valitsi kriteerit. Edellä mainittujen menetelmien lisäksi esimerkiksi typologinen analyysi (Cohen ym. 2007, 473) voisi olla mahdollinen menetelmä. Varsinaista valintaa ei kuitenkaan ole mahdollista tehdä ennen kuin kriteerit ovat selvillä ja analysoitavan tiedon laatu siis tiedossa.

6 Tutkimuksen suorittaminen

Vertailuja varten testattavia järjestelmiä haluttiin kokeilla käytännössä, jotta lopputulos ei olisi riippuvainen pelkästään ulkopuolisten toimijoiden kertomista asioista. Tätä varten järjestelmät on asennettava ja niitä on kokeiltava olemassa olevan rajapinnan testaamiseen. Eri järjestelmien asentaminen samalle tietokoneelle voi olla haastavaa, koska ne saattavat kilpailla samoista tietokoneen resursseista. Tämä voidaan välttää käyttämällä virtualisointijärjestelmää kuten Docker, joka on tutkimuksen tekijälle ennestään tuttu. Näin jokainen sovellus voidaan asentaa omaan virtuaaliseen tietokoneeseensa jotka voivat kuitenkin kommunikoida keskenään virtualisointijärjestelmän sisäisen lähiverkon kautta.

Testien kohdejärjestelmän, joka tarjoaa kokeisiin tarvittavan rajapinnan, ei tarvitse olla tuotanto-käytössä, kunhan sen tiedetään toimivan odotetusti. Näin voidaan luottaa, että testeissä esiin tulevat ongelmat johtuvat testin kohteena olevasta testiautomaatiojärjestelmästä eikä kohdejärjestelmästä. Tähän tutkimukseen kohdejärjestelmäksi valitaan Spring-sovelluskehyksen RESTful Web Service-esimerkki (Building a RESTful Web Service 2017). Esimerkin yksinkertaisuuden ja Spring-kehyksen laajan käytön vuoksi esimerkin voidaan olettaa kohtuudella toimivan oikein.

6.1 Vertailukriteerien ja mittarien valinta

Jotta järjestelmiä voitaisiin vertailla mahdollisimman objektiivisesti, on tärkeää, että vertailukriteerit ja vaadittavat ominaisuudet valitaan ennen vertailuun otettavia järjestelmiä. Lisäksi kriteerit on pyrittävä perustelemaan todellisilla kohdeyrityksen tarpeilla tai tuotantoympäristön ominaisuuksilla.

Järjestelmän suorituskyvyn mittaamista esimerkiksi testeihin kuluvana aikana ei pidetty tarkoituksenmukaisena. Koska testit on tarkoitus käynnistää ja niiden tulokset raportoida automaattisesti, ja testit suorittaa väsymätön kone, ei itse prosessiin kuluvan ajan tarkalla määrällä ole merkitystä. Mahdollisesti jumiin jäävät testiprosessit huomataan ja korjataan kohdeyrityksen järjestelmien normaalin valvonnan ja ylläpidon kautta.

Näiden rajausten jälkeen yhtään määrälliseen analyysiin soveltuvaa mittaria ei tullut esiin. Kaikki havainnot analysoitiin siis laadullisin menetelmin.

6.1.1 Sopivuus kohdeyrityksen infrastruktuuriin

Kohdeyritys pyrkii käyttämään ensisijaisesti Linux-pohjaisia käyttöjärjestelmiä. Kaikki tuotteet ja tuotannon tukijärjestelmät, joita ei ole hankittu SaaS-periaatteella, ylläpidetään Linux-pohjaisilla palvelimilla. Kehitystyössä käytettäviin työasemiin suositellaan myös Linux-käyttöjärjestelmää, jotta kehittäjät voisivat kokeilla tekemiään muutoksia ja etsiä virheitä suorittamalla sovellusta omalla työasemallaan.

Siksi valittavan testiautomaatiojärjestelmänkin tulee toimia Linux-käyttöjärjestelmällä. Tämä kriteeri ehdottomasti sulkee pois kaikki vaihtoehdot joista ei ole Linux-versiota saatavilla. Järjestelmää jota ei voida kokeilla Linuxissa ei voida edes harkita käyttöönotettavaksi.

6.1.2 Testitapausten laatiminen ja ylläpito

Tutkimuksen tarkoitus on löytää järjestelmä, joka paitsi tuottaa kohdeyritykselle hyötyä vähentämällä kehitystyössä syntyviä virheitä ei myöskään tarpeettomasti kuormita yrityksen työntekijöitä käyttöönoton ja käytön aikana. Testitapausten määrittely, toteuttaminen ja ylläpito tulevat projektien kehittäjien muiden tehtävien rinnalle. Siksi niissä onnistuminen ei saa vaatia merkittävää määrää uusien ohjelmointikielien tai ohjelmistojen opiskelua.

Jotta järjestelmiä voitaisiin vertailla objektiivisesti, tämä kriteeri päätettiin luokitella kolmeen tasoon:

1. Helppo – Testit voidaan toteuttaa jo käytössä olevilla ohjelmointikielillä ja -sovelluksilla, eikä kehittäjien tarvitse käyttää merkittävästi aikaa tiedonhankintaan tai opiskeluun.
2. Kohtuullinen – Kehittäjät tarvitsevat joitakin uusia tietoja tai taitoja, jotka he voivat itsenäisesti opiskella verkossa käyttäen esimerkiksi tunnin silloin tällöin muiden töiden lomassa.
3. Vaikea – Kehittäjät tarvitsevat ulkopuolisen kouluttajan tai kokonaisia työpäiviä tarvittavien tietojen ja taitojen itsenäiseen opiskeluun.

6.1.3 Integroituminen kohdeyrityksen muihin järjestelmiin

Kohdeyrityksessä on käytössä Jenkins CI-järjestelmä. Integroituminen tähän tarkoittaa, että joko Jenkins tai testijärjestelmä luo testattavasta järjestelmästä asennuksen, jota vastaan testijärjestelmä ajaa testit. Olisi suotavaa, että valittava järjestelmä voitaisiin käynnistää näin, jotta testien ajaminen ei olisi sen varassa, että työntekijä muistaa käynnistää ne tehtyään muutoksia kehitettävään järjestelmään.

Kohdeyritys käyttää tehtävien hallintaan ja seurantaan Atlassian JIRA-järjestelmää. Olisi toivottavaa, että valittava järjestelmä voisi luoda virheeseen päätyvistä testitapauksista

automaattisesti tehtävät tähän järjestelmään. Näin virheraporttien seuraaminen ja virheiden päätyminen korjattavien asioiden listalle ei jäisi ihmisten varaan.

Vertailun objektiivisuuden varmistamiseksi tämän kriteerin arvioinnissa päätettiin käyttää kolmea tasoa:

1. Hyvä – Järjestelmä voidaan integroida molempiin haluttuihin järjestelmiin ilman merkittävää työpanosta tai kustannuksia.
2. Kohtuullinen – Toinen järjestelmä voidaan integroida ilman lisäpanostusta mutta toisen integrointi ei onnistu, vaatii työtä tai lisää kustannuksia.
3. Riittämätön – Kumpaakaan järjestelmää ei saada integroitua ollenkaan tai ilman merkittävää työmäärää tai rahallista panostusta.

6.1.4 Tuki REST-rajapintojen testaamiselle

Jotta järjestelmällä voidaan testata REST-rajapintaa, on testitapausten kyettävä avaamaan yhteys REST-rajapintaan, lähettämään kysely ja ottamaan vastaan rajapinnan antama vastaus. Mikäli järjestelmässä ei itsessään ole tähän kykeneviä toimintoja on siihen saatava nämä toiminnot mahdollistava lisäosa.

Vertailun objektiivisuuden parantamiseksi päätettiin tämä kriteeri luokitella seuraaviin tasoihin:

1. Hyvä – Kaikki tarvittavat ominaisuudet ovat järjestelmässä valmiina tai se lataa tarvittavat lisäosat automaattisesti tai pienen konfiguraatiomuutoksen jälkeen.
2. Kohtuullinen – Työntekijän täytyy erikseen ladata ja asentaa tarvittavat lisäosat ennen kuin ominaisuuksia voidaan käyttää testitapauksissa.
3. Riittämätön – Kaikkia tarvittavia ominaisuuksia ei ole järjestelmään saatavana ainakaan ilman merkittävää rahallista panostusta.

6.1.5 Sopivuus kohdeyrityksen organisaatioon

Kohdeyritys toimii varsin itsenäisinä tiimeinä ja projekteina. Vaikka tavoite on standardisoida projekteissa käytettävät työkalut, halutaan projektien tiedot pitää itsenäisinä ja erillisinä. Erillisen palvelimen pystyttäminen jokaista projektia varten ei ole kannattavaa, kun siitä syntyvien suorien kustannuksien lisäksi nämä pitää lisätä yrityksen valvontajärjestelmiin ja infrastruktuurilistoihin. Erityisen suuriksi hyötyyn nähden nämä kustannukset kasvaisivat, mikäli palvelimet pitäisi pystyttää uudestaan, kun johonkin projektiin joudutaan palaamaan pidemmän ajan kuluttua esimerkiksi sittemmin löydetyn virheen korjaamiseksi.

Tälle kriteerille ei löydetty sopivia luokituksia joiden kautta sitä voitaisiin arvioida. Siksi sen käsittelyssä on yleisessä tapauksessa kiinnitettävä erityistä huomiota objektiivisuuden säilyttämiseen tutkimuksessa, jotta tutkimuksen validiteetti ei vaarantuisi. Tässä tutkimuksessa

tutkija kuitenkin samalla edustaa kohdeyrityksen organisaatiota, joten mahdollinen subjektiivinen bias ei ole vahingollinen tutkimuksen lopputulokselle.

6.2 Tutkittavien järjestelmien valinta

Sopivien järjestelmien löytämiseksi tehtiin verkosta hakuja. Käytetyt hakutermit suomennoksineen näyttää Taulukko 1. Hauista nousi konsistentisti esille yksi varsin tuore lähde, joka esitteli kuusi testijärjestelmää lyhyesti. Tämän lähteen järjestelmiin tutustuttiin tarkemmin ja niistä valittiin 3 järjestelmää koeasennuksiin. Tutkimuksen ulkopuolelle rajattiin järjestelmät, joiden kuvattiin tarkoitetun lähinnä käyttöliittymän testaamiseen tai olivat vielä kehitystyönsä alkuvaiheessa. (Colantonio 2016.)

6.2.1 Tutkimukseen valitut järjestelmät

Serenity on useampien testiautomaatiojärjestelmien kanssa toimiva kirjasto. Kohdeyrityksessä on jo käytössä alun perin Java-ohjelmointikielen yksikkötestaukseen tarkoitettu jUnit-järjestelmä. Serenity laajentaa jUnit:a toiminnallisten testitapausten suorittamiseen ja yksityiskohtaisempaan raportointiin. Se myös integroituu syvästi REST Assured-kirjastoon REST-rajapintojen testaamiseen. (Colantonio 2016.)

Robot Framework on alun perin Nokia Networksin omaan käyttöön kehittämä testiautomaatiojärjestelmä. Se on myöhemmin julkaistu avoimen lähdekoodin lisenssin alla ja saavuttanut melko suuren suosion testiautomaation toteuttajien keskuudessa. Robot Frameworkin testitapaukset kirjoitetaan omalla avainsanojen tunnistukseen perustuvalla kielellään ja ne voidaan upottaa esimerkiksi HTML-sivulle vaikkapa määrittelydokumentin sisään. (Colantonio 2016.)

Hakutermi	Suomennos
Test Automation	Testiautomaatio
Regression Testing	Regressiotestaus
REST	REST (akronyymille ei ole vakiintunutta suomennosta)
Open Source	Avoin lähdekoodi

Taulukko 1: Järjestelmien etsimisessä käytetyt hakutermit.

RedwoodHQ on uudempi testiautomaatiojärjestelmä, jonka suunnittelussa on painotettu helppoa käyttöliittymää. Se vaatii keskitetyn asennuksen ja agenttiohjelman jokaiselle koneelle,

jolla testejä halutaan suorittaa. Siksi se sopii paremmin isompien ja keskitetymmin johdettujen organisaatioiden kuin pienempien itseohjautuvien tiimien käyttöön. (Colantonio 2016.)

6.2.2 Tutkimuksesta pois jätetyt järjestelmät

Sahi on verkkosovellusten käyttöliittymien testaamiseen tarkoitettu järjestelmä. Sen testitapaukset tehdään käyttämällä sovellusta tavalliseen tapaan samalla kun Sahi tallentaa käyttäjän toimet, syöttämät tiedot ja järjestelmän vastaukset. Testitapauksia voi jälkikäteen muokata ja lisätä niihin logiikkaa, mutta tällä menetelmällä ei ole mahdollista testata teknistä rajapintaa jolla ei ole käyttöliittymää. (Colantonio 2016.)

Galen Framework on nimenomaan käyttäjäkokemuksen testaamiseen tarkoitettu järjestelmä. Sen määrittelykieli mahdollistaa verkkosivujen käytön ja halutun palautteen määrittelyn yksityiskohtaisesti, mutta Sahin tapaan testit suoritetaan selainympäristössä. Sekään ei siis pysty testaamaan käyttöliittymättömiä rajapintoja. (Colantonio 2016.)

Gauge on uusi tulokas testiautomaation alalla ja sitä julkaistaan toistaiseksi beta-statuksen alla. Beta-status tarkoittaa, että julkaisijat eivät pidä sovellusta valmiina eikä se välttämättä sovellu vielä tuotantokäyttöön. Gaugen testitapaukset kirjoitetaan Robot Frameworkin tapaan omalla luonnollista kieltä muistuttavalla kielellään, mutta ne eivät ole samalla tavalla upotettavissa dokumentaation sekaan. Beta-statuksen vuosi Gauge päätettiin tässä kohdassa rajata testien ulkopuolelle. (Colantonio 2016.)

6.3 Koeasennukset

Koeasennukset suoritettiin asentamalla kokeiden kohdejärjestelmä ja kukin testattavista järjestelmistä omaan virtuaaliseen tietokoneeseen, joita ajettiin yhdellä fyysisellä tietokoneella. Näin saatiin aikaan kontrolloitu, toistettava ja jokaiselle testattavalle järjestelmälle identtinen suoritus-ympäristö joka vastasi riittävässä määrin kohdeyrityksen tuotantotoiminnassaan käyttämiä järjestelmiä.

Koeasennuksia tehtäessä laadittiin muistiinpanoja järjestelmien hyvistä ja hankaliksi koetuista puolista muistiinpanoja. Muistiinpanojen avulla järjestelmien subjektiivisia ominaisuuksia voitiin arvioida myöhemmin, kun kaikki koeasennukset oli suoritettu.

7 Havainnot ja tulokset

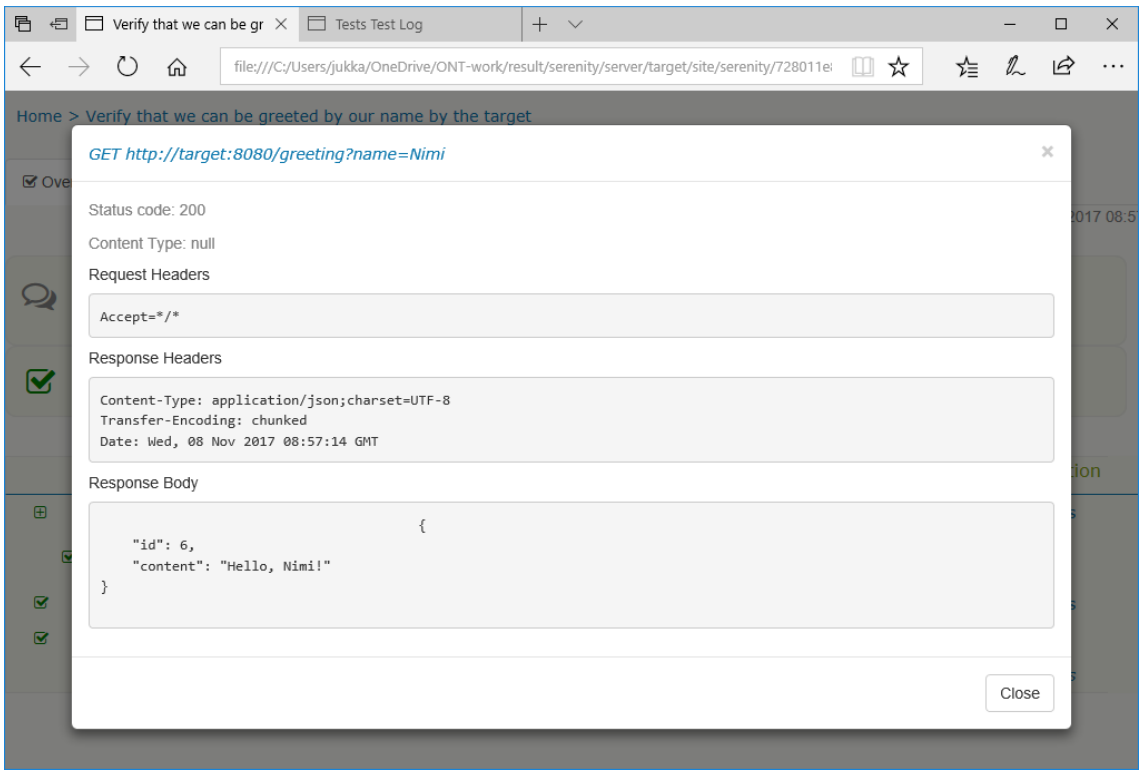
Koeasennuksista saatiin selkeä käsitys järjestelmien sopivuudesta kohdeyrityksen käyttöön. Järjestelmistä löydettiin myös eroavaisuuksia, joita vertailukriteereitä asetettaessa ei osattu ottaa huomioon. Näitä eroavaisuuksia on käsitelty järjestelmistä tehtyjen havaintojen käsittelyn jälkeen.

7.1 Serenity

Serenity on kirjoitettu Java-ohjelmointikielellä, joka on alkujaankin tarkoitettu käyttöjärjestelmä-agnostiseksi ohjelmistokehitystyökaluksi. Kohdeyrityksen tärkeimmät tuotteet ovat myös kirjoitettu Javalla, joten Serenityn toimivuus kohdeyrityksen infrastruktuurissa on erittäin hyvä.

Serenityn testitapaukset kirjoitetaan Java-ohjelmointikielellä ja ne suoritetaan jUnit-testikehyksellä, joka on kohdeyrityksessä jo käytössä yksikkötestauksessa. Liitteessä 1 on esimerkki Serenityn testitapauksen koodista. Kohdeyrityksessä käytössä oleva Maven-käännösjärjestelmä lataa halutut lisäosat automaattisesti, kun ne on vain merkitty käytettäviksi. Serenityn käyttöönotto ei siksi vaadi kehittäjiltä erityisiä uusien taitojen opiskelua tai uusien sovellusten asentamista ja opettelemista.

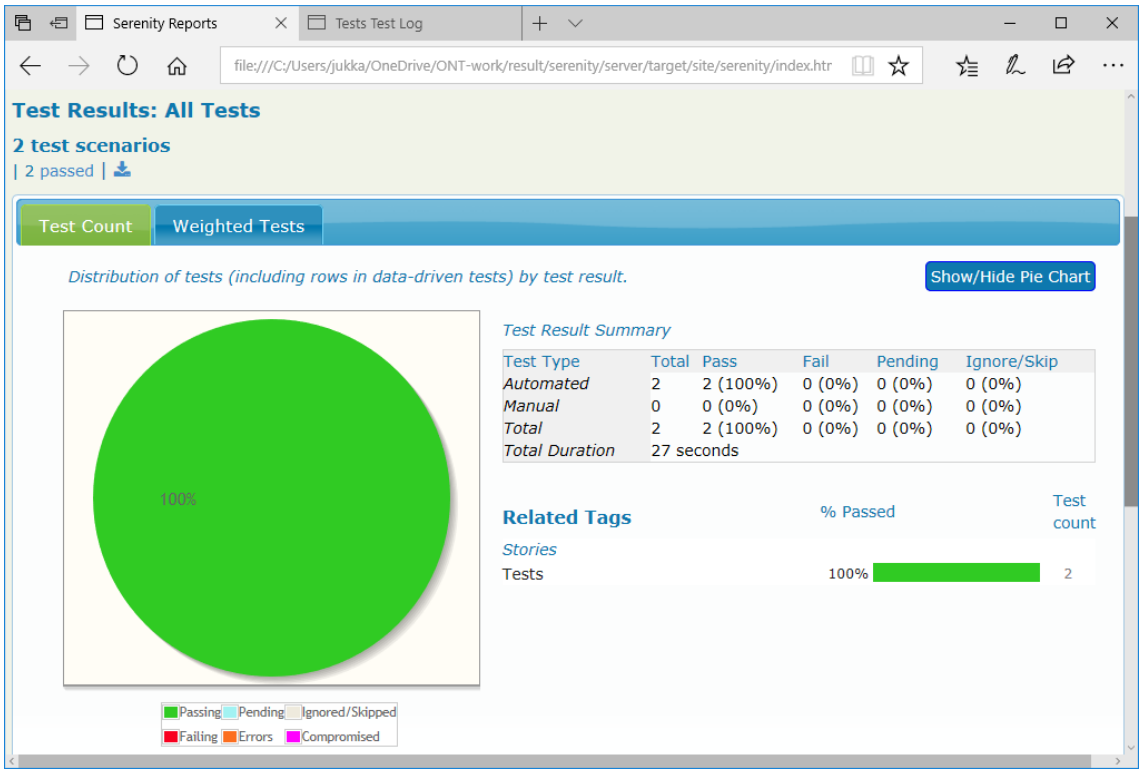
Koska Serenityn testit suoritetaan jo käytössä olevien järjestelmien kautta ovat ne jo valmiiksi integroituja Jenkins CI-järjestelmään. Testien linkittämiseen JIRA-tiketteihin on myös olemassa valmis helppokäyttöinen lisäosa. REST-rajapintojen testaamiseen Serenity tukee Rest Assured-kirjastoa, jota se vielä laajentaa niin että testiraportilta voi katsoa kirjaimelleen mitä sanomia rajapinnan ja testiasiakkaan välillä on liikkunut, kuten kuvio 3 näyttää.



Kuvio 3: Serenity tallentaa testiaskeleista täydelliset tiedot.

Testiraportin tuottamisessa oli kuitenkin yksi hankala kohta, kun Serenity ei aluksi tuottanut sitä ollenkaan, mikäli jokin testeistä epäonnistui. Ongelma johtui Mavenin konfiguraatiosta, joka ei sallinut prosessin jatkua epäonnistuneen testin jälkeen. Tämä osoittaa kuinka tiiviisti yhteen integroiduissa järjestelmissä voi olla yllättäviä riippuvuuksia, eikä ongelmien syy aina ole intuitiivisesti löydettävissä. Tämä ongelma ei osunut mihinkään valittuihin kriteereihin, eikä sitä pidetty kriittisenä. Kun ongelma on nyt kerran ratkaistu, niin ratkaisu on tiedossa ja helposti toistettavissa mikäli siihen törmätään uudestaan.

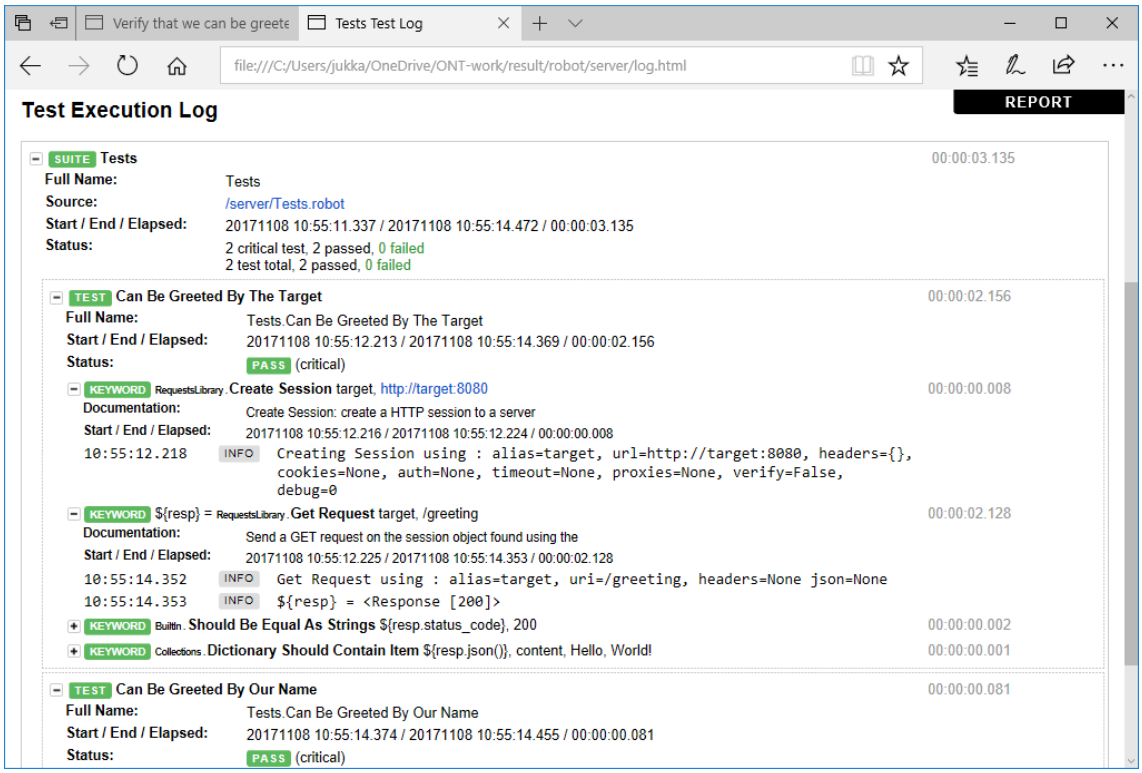
Serenity integroituu saumattomasti kohdeyrityksen nykyiseen prosessiin, jossa projektikohtaisilla asetuksilla ja projektin tietojen pitämisellä mahdollisimman tiivisti yhtenä pakettina on suuri painoarvo. Siksi se sopii erinomaisesti kohdeyrityksen organisaatioon. Kuvio 4 näyttää Serenityn raportin etusivun.



Kuvio 4: Serenityn raportissa on selkeän yleiskuvan antava piirakkakaavio.

7.2 Robot Framework

Robot Framework löytyy käytössä olevien Linux-jakelujen normaaleista sovellusvalikoimista. Se on siten helposti asennettavissa ja jakelijat pyrkivät varmistamaan sen toimivuuden. Robot Frameworkin vakiokirjastoista löytyy verkkoyhteyksien käyttöön tarkoitettu RequestsLibrary. Se ei kuitenkaan tarjoa erityisiä ominaisuuksia rajapinnan antamien vastauksien oikeellisuuden tarkistamiseen. Kuvio 5 näyttää kuinka Robot Framework esittää verkkoliikenteen tiedot testiraportissa.



Kuvio 5: Robot Framework näyttää testiaskelten yksityiskohdat listan sisällä.

Robot Frameworkin testitapaukset kirjoitetaan luonnollista kieltä mukailevalla kuvauskielellä, ja ne voidaan myös upottaa dokumentaation sisään. Testitapaukset voidaan tallentaa projektin muun lähdekoodin yhteyteen versionhallintaan. Näin niiden ylläpito muiden projektiin tehtävien töiden mukana onnistuu jouhevasti. Kieli on suhteellisen yksinkertainen, mutta vaatii kuitenkin opettelua ja käytössä olevien kehityssovellusten tuki tämän kielen käyttöön ei ole kovin laaja. Esimerkki tällä kielellä kirjoitetusta testitapauksesta on liitteessä 2. Huomaa että testitapauksessa myös sanojen välissä olevien välilyöntien määrällä on merkitys. Tämä voi hämmentää kuvauskieltä ennestään tuntematonta henkilöä.

Jenkins CI-järjestelmä on helposti asetettavissa käynnistämään Robot-testit käännösprosessin osana. Sen sijaan testitulosten viemiselle JIRA-järjestelmään ei löydy sopivaa lisäosaa. JIRAan itseensä on kyllä saatavana maksullinen lisäosa joka osaa lukea Robot Frameworkin tuottamia raporttitiedostoja, kuten osaa myös Atlassianin Jenkinsiä vastaava Bamboo CI-järjestelmä. Kumpikin vaihtoehto kuitenkin tuottaisi kohdeyritykselle lisäkustannuksia.

7.3 RedwoodHQ

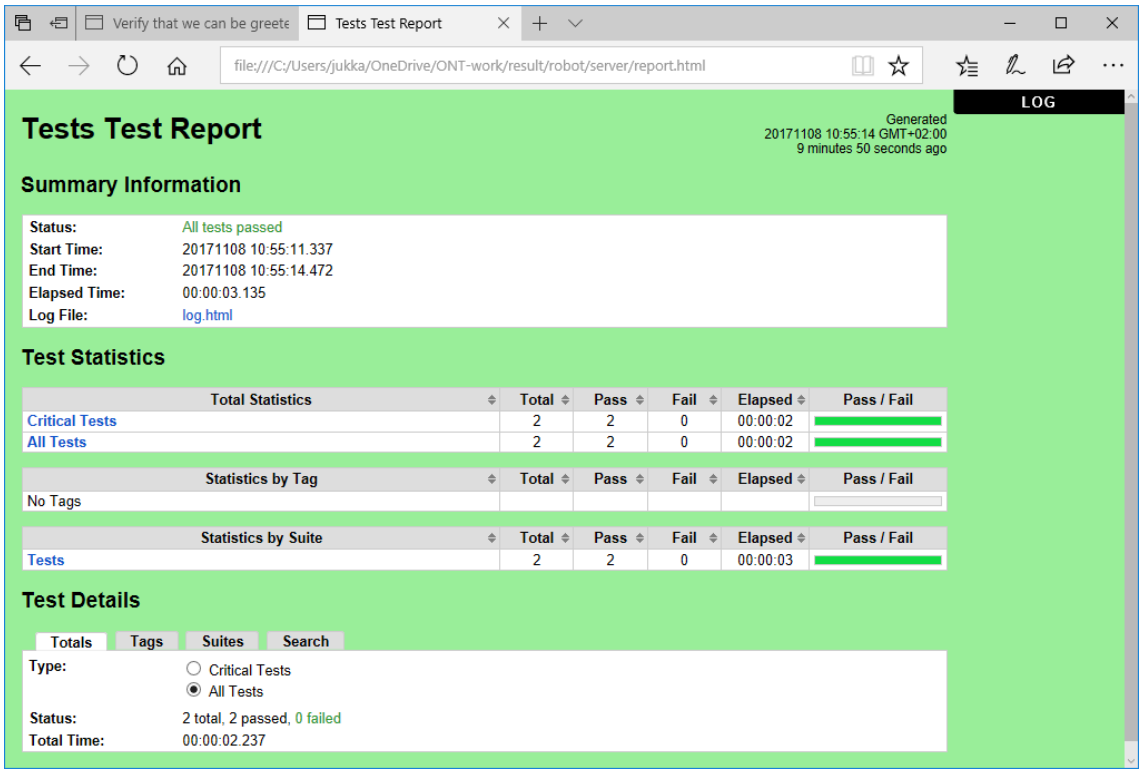
RedwoodHQ:n Linux-version asennuspaketti aiheutti asennuksen keskeyttävän virheen. Verkosta tehdyssä haussa ilmeni, että myös toisella taholla oli tapahtunut sama virhe asennuspaketin kanssa. Järjestelmän kehittäjät eivät olleet vastanneet virheestä tehtyyn ilmoitukseen

kahden kuukauden aikana, ja projektin versionhallintasivulta nähtiin, että vaikka kehitystyötä tapahtuu, se on melko verkkaista. (Dominik 2017.)

Julkaisupakettiin jääneen vian ja hitaan kehitystyön johdosta arvioitiin, että RedwoodHQ:n kehitys ei ole riittävän aktiivista takaamaan kohdeyrityksen toivomaa luotettavuutta ja tukea sen käyttämisessä. Näiden syiden vuoksi RedwoodHQ pudotettiin pois tutkimuksesta tässä kohdassa eikä sitä harkittu testattavaksi esimerkiksi toisella käyttöjärjestelmällä.

7.4 Odottamaton havainto

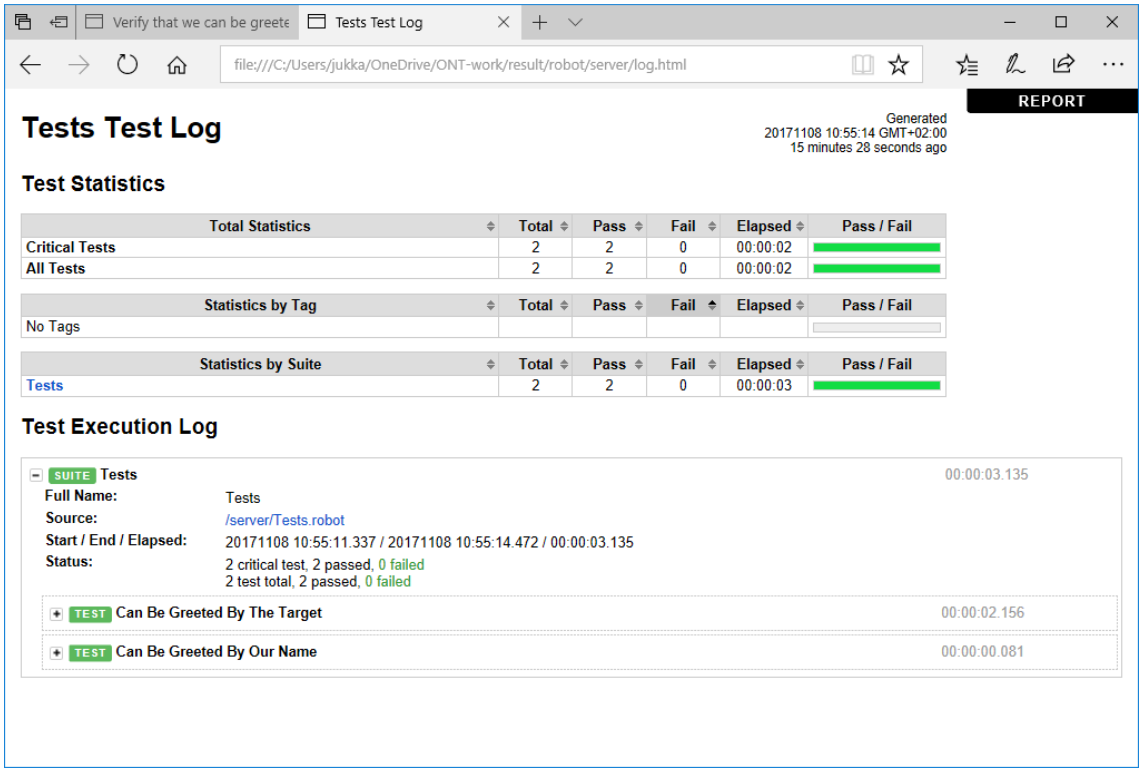
Kokeiden aikana ilmeni, että järjestelmien tuottamissa raporteissa on merkittäviä eroja. Molemmat testatut järjestelmät tuottavat HTML-muotoisen sivukokoelman, jota selaamalla eri testien tuloksia ja kulkua voi tutkia. Kuvio 2 näyttää esimerkin Serenityn testiraportin etusivusta. Sivulla on iso selkeä piirakkakaavio, joka näyttää testien kokonaistilanteen, sekä vähän tarkempia tietoja testien tilasta. Kuvio 6 näyttää Robot Frameworkin raportin etusivun, jolla paljon enemmän yksityiskohtaisempaa tietoa. Sen sijaan yleiskuvan saaminen on hitaampaa, koska yhtä selkeää elementtiä joka tämän tiedon antaisi ei löydy.



Kuvio 6: Robot Framework lajittelee testitapaukset useilla eri tavoilla.

Robot Framework kokoaa kaikkien testien lokitiedot yhdelle sivulle, kuten kuvio 7 näyttää. Tällä sivulla myös toistuu osa etusivun mittareista. Kuvio 5 näyttää kuinka Robot Framework esittää testiaskeleiden yksityiskohtaiset tiedot lokin sisällä. Serenity luo jokaiselle testille

oman lokisivun. Kuvio 2 on esimerkki yhdestä Serenityn lokisivusta ja kuvio 3 näyttää kuinka yksityiskohtaiset tiedot esitetään pop up-ikkunassa. Merkillepantavaa on, että Serenity on tallentanut verkkoliikenteen täydellisen sisällön, kun Robot Frameworkin logista löytyy vain joitakin metatietoja, kuten kuvio 5 näyttää. Tällä voi olla huomattava merkitys testin löytämän virheen korjaamisessa, kun kehittäjä voi tarkistaa suoraan lokista mitä sanomia testijärjestelmän ja kohdejärjestelmän välillä on liikkunut.



Kuvio 7: Robot Frameworkin lokisivulla toistuvat etusivun mittarit.

7.5 Tulokset

Testatut järjestelmät täyttivät asetetut kriteerit yleisesti ottaen erittäin hyvin. Koska RedwoodHQ:ta ei kyetty asentamaan, ei sitä voitu kokeilla useimpien kategorioiden kohdallakaan. Tämä koettiin hieman harmilliseksi, sillä nyt testatut järjestelmät olivat hyvin saman tyyppiset eikä niiden välille löytynyt suuria eroja.

Taulukko 2 esittää koeasennusten tulokset kootussa muodossa. RedwoodHQ merkittiin sopivaksi järjestelmiin asennusvirheestä huolimatta, sillä Linux-versio oli kuitenkin tarjolla. Serenity ja Robot Framework olisivat molemmat ominaisuuksiensa puolesta käyttökelpoisia kohdeyritykselle. Serenity kuitenkin saavutti paremman luokituksen testitapausten laatimisessa ja integroitavuudessa, ja on suositeltavin valinta.

8 Yhteenveto ja johtopäätökset

Tutkimuksessa vertailtiin avoimen lähdekoodin testiautomaatiojärjestelmien sopivuutta kohdeyrityksen käyttöön. Ensin laadittiin kriteeristö, jonka avulla järjestelmiä voitiin vertailla mahdollisimman objektiivisesti. Osa kriteereistä tarkasteli järjestelmien teknistä yhteensopivuutta kohdeyrityksen olemassa olevan infrastruktuurin kanssa. Loput kriteerit mittasivat kuinka järjestelmät sopivat kohdeyritykselle organisaationa.

	Serenity	Robot Framework	RedwoodHQ
Sopivuus järjestelmiin	Sopii	Sopii	Sopii
Testitapausten laatiminen	Helppo	Kohtuullinen	?
Integroitavuus	Hyvä	Kohtuullinen	?
REST-tuki	Hyvä	Hyvä	?
Sopivuus organisaatioon	Hyvä	Hyvä	?

Taulukko 2: Järjestelmien arviointi kriteerien perusteella. RedwoodHQ:ta ei pystytty kokeilemaan.

Seuraavaksi etsittiin verkosta hakupalvelujen avulla sopivalta vaikuttavia testiautomaatiojärjestelmiä. Löydösten perusteella valittiin kolme järjestelmää lähempään tarkasteluun. Näistä yksi kuitenkin jouduttiin hylkäämään alkuvaiheessa viallisen asennuspaketin vuoksi.

Kaksi jäljellejäänyttä järjestelmää asennettiin koeluontoisesti kumpikin omalle virtuaaliselle palvelimelleen. Niille laadittiin myös lyhyet testitapaukset, joiden kohteena oli Spring-sovelluskehyksen esimerkkirajapinta. Näiden koeasennusten myötä tehtiin havaintoja, joita analysointiin laadittujen kriteerien kautta.

Molemmat onnistuneesti testatut järjestelmät sopisivat kohdeyrityksen käyttöön, mutta toinen niistä täytti määritellyt kriteerit paremmin kuin toinen. Kohdeyritykselle annetaan suositus ottaa REST-rajapintojen regressiotestauksen automatisoinnissa käyttöön Serenity.

Järjestelmän sopivuudelle organisaatioon ei kyetty määrittämään selkeää luokittelua. Kuitenkin molempia järjestelmiä, joita kyettiin testaamaan, pidettiin yhdenvertaisina tässä suhteessa. Näin ollen lopputulos ei muuttuisi, vaikka tämä kriteeri jätettäisiin pois vertailusta. Siten tämä ongelma ei itse asiassa vaikuta tutkimuksen validiteettiin.

Testiautomaatiojärjestelmien tuottamien raporttien sisällöissä huomattiin eroja, joilla todennäköisesti on vaikutusta järjestelmien löytämien virheiden korjaamiseen kuluvaan aikaan. Tällaisten huomioimattomien tekijöiden läsnäolo voi heikentää tutkimuksen validiteettia, jos toinen järjestelmä olisi tullut valituksi, mikäli tämä ominaisuus olisi sisällytetty tutkimuksen kriteereihin. Tässä tapauksessa kuitenkin valittu järjestelmä tuotti yksityiskohtaisinta tietoa, joten tutkimusta ei päätetty toistaa täydennetyin kriteerein.

Reliabiliteetti on tutkimuksen kannalta hankala konsepti, koska sekä testattavat järjestelmät että kohdeyritys tarpeineen ovat kehittyviä uniikkeja yksiköitä. Sen vuoksi voi tutkimuksen tulos toisena ajankohtana tai eri kohdeyrityksen kohdalla olla eri, vaikka se toistettaisiinkin täydellisesti. Sen sijaan valittujen kriteerien valossa voidaan olettaa, että eri tutkija olisi näistä havainnoista päätynyt samaan lopputulokseen, sillä yksi ja vain yksi vertailluista järjestelmistä saavutti parhaan luokan kaikilla kriteereillä.

9 Jatkotutkimusehdotukset ja parannusehdotukset

Työkalun käyttöönoton vaikuttavuutta olisi hyödyllistä seurata, jotta saadaan tietoa sen tuomista hyödyistä tai työkalusta voidaan luopua, jos hyötyä ei saadakaan. Saatavan hyödyn ei tarvitse olla taloudellista, vaan se voi näkyä myös esimerkiksi parantuneena asiakastyytyväisyytenä. Tämän seurannan voisi toteuttaa esimerkiksi keräämällä tietoa mikä osuus havaituista virheistä koskee REST-rajapintojen toimintaa ennen ja jälkeen työkalun käyttöönoton ja missä vaiheessa tuotantoprosessia virheet on huomattu. Näitä havaintoja pitäisi sitten verrata projektien kustannuksiin ja asiakastyytyväisyyskyselyjen tuloksiin.

Tutkimuksessa ei käytetty kriteerinä järjestelmien tuottamien raporttien sisältämiä tietoja. Tutkimuksen aikana kuitenkin huomattiin, että näissä on eroja, joilla todennäköisesti on vaikutusta työkalun löytämän virheen korjaamiseen kuluvaan aikaan. Tämä on syytä ottaa huomioon myöhemmissä testiautomaatiojärjestelmien valintaa koskevissa tutkimuksissa.

Lähteet

Painetut

Cohen, L., Morrison, K. & Manion, L. 2007. Research Methods in Education. 6. painos. Abingdon: Routledge.

Jia, X. 2000. Object-oriented software development in Java: principles, patterns, and frameworks. Reading: Addison Wesley Longman.

Kasurinen, J. P., 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Mathur, A. P. 2008. Foundations of Software Testing. Delhi: Dorling Kindersley (India).

Richardson, L. & Ruby, S. 2007. RESTful Web Services. Sebastopol: O'Reilly.

Sähköiset

Apache Maven Project. 2017. Mavenin esittely. Viitattu 23.11.2017. https://maven.apache.org/what-is-maven.html

Building a RESTful Web Service. 2017. Pivotal. Viitattu 20.10.2017. https://spring.io/guides/gs/rest-service/

Colantonio, J. 2016. 6 top open-source testing automation frameworks: How to choose. Viitattu 20.10.2017. https://techbeacon.com/6-top-open-source-testing-automation-frameworks-how-choose

Dominik. 2017. tar: Exiting with failure status due to previous errors - RedWoodHQ installation issue. Keskusteluryhmäkirjoitus. Viitattu 2.11.2017. https://groups.google.com/d/msg/primatest-automation/P9KoCgmqV-Y/EnziR60JBgAJ

Mitä vapaat ohjelmistot ovat? 2017. Free Software Foundation Europe. Viitattu 2.11.2017. https://fsfe.org/about/basics/freesoftware.fi.html

Eduix. 2009. JIRAn esittely. Viitattu 22.11.2017. http://www.eduix.fi/9-tuotteet-ja-palvelut/9-jira

Mikä on SaaS? 2010. Web-opas. Viitattu 22.11.2017. http://www.webopas.net/saas.html

Paketinhallintajärjestelmä. 2009. Linux.fi. Viitattu 22.11.2017. https://www.linux.fi/wiki/Paketinhallintajärjestelmä

Poimala, S. & Tolvanen, P. 2013. Ketteryys haltuun: Yleisimmät ketterät käytännöt. Viitattu 23.11.2017. https://www.meteoriitti.com/2013/06/06/ketteryys-haltuun-yleisimmat-ketterat-kaytannot/

What is Docker? 2014. Opensource.com. Viitattu 23.11.2017. https://opensource.com/resources/what-docker

XML Soap. 2003. W3Schools. Viitattu 22.11.2017. https://www.w3schools.com/xml/xml_soap.asp

Julkaisemattomat

Manninen, J.-P. 2015. Teemasta opinnäytetyöksi. Oppimistehtävä. Laurea-ammattikorkeakoulu. Kerava.

N.N. 2015. Testauspäällikön haastattelu 17.9.2015.

Kuviot

Taulukot

Liitteet

Liite 1: Esimerkki Serenityn testitapauksen koodista.

```
package tests;

import io.restassured.RestAssured;
import io.restassured.response.Response;
import net.serenitybdd.junit.runners.SerenityRunner;
import net.serenitybdd.rest.SerenityRest;
import net.thucydides.core.annotations.Steps;
import net.thucydides.core.annotations.Step;
import org.junit.Test;
import org.junit.runner.RunWith;

import static org.hamcrest.Matchers.is;

public class TestSteps {
   private Response response;

   @Step
   public TestSteps whenGreeting() {
      response = SerenityRest.when()
                             .get("http://target:8080/greeting");
      return this;
   }

   @Step
   public TestSteps whenGreetingFor(String name) {
      response = SerenityRest.when()
                             .get("http://target:8080/greeting?name=" + name);
      return this;
   }

   @Step
   public TestSteps statusCodeIs(int statusCode) {
      response.then()
              .statusCode(statusCode);
      return this;
   }

   @Step
   public TestSteps contentIs(String content) {
      response.then()
              .body("content", is(content));
      return this;
   }
```

```
}

@RunWith(SerenityRunner.class)
public class Tests {
   @Steps
   TestSteps steps;

   @Test
   public void verifyThatWeCanBeGreetedByTheTarget(){
      steps.whenGreeting().statusCodeIs(200)
                          .contentIs("Hello, World!");
   }

   @Test
   public void verifyThatWeCanBeGreetedByOurNameByTheTarget(){
      steps.whenGreetingFor("Nimi").statusCodeIs(200)
                                   .contentIs("Hello, Nimi!");
   }
}
```

Liite 2: Esimerkki Robot Framework-testitapauksen koodista.

```
*** Settings ***

Library  Collections
Library  RequestsLibrary

*** Test Cases ***

Can Be Greeted By The Target
    Create Session  target  http://target:8080
    ${resp}=  Get Request  target  /greeting
    Should Be Equal As Strings  ${resp.status_code}  200
    Dictionary Should Contain Item  ${resp.json()}  content  Hello, World!

Can Be Greeted By Our Name
    Create Session  target  http://target:8080
    ${params}=  Create Dictionary  name=Nimi
    ${resp}=  Get Request  target  /greeting  params=${params}
    Should Be Equal As Strings  ${resp.status_code}  200
    Dictionary Should Contain Item  ${resp.json()}  content  Hello, Nimi!
```